AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of increasing the execution speed of invoking and returning from a Method of a plurality of Methods executing on a computer system while reducing the supporting memory footprint, the method comprising:

establishing ~~an activation stack~~ a single frame size determining template ~~with~~ for comparing the single frame size determining template with each of the Methods of the plurality of Methods, the single frame size determining template having a set of frame size determining criteria representative of a predetermined number of words for determining the size of activation frames when creating the activation frames;

determining whether a word number requirement of the Method conforms to the frame size determining criteria of the ~~stack~~ single frame size determining template;

conditionally creating a fixed size activation frame regardless of ~~the Method's~~ an exact stack requirement[s] of the Method, based on the set of frame size determining criteria of the ~~activation stack~~ single frame size determining template if the word number requirement of the Method conforms to the set of frame size determining criteria of the ~~activation stack~~ single frame size determining template;

conditionally creating an activation frame to match the Method's exact stack requirements if the word number requirement of the Method does not conform to the set

of frame size determining criteria of the ~~activation stack~~ single frame size determining template; ~~and~~

spatially optimizing the Method to provide a Method access structure; and

associating [a] the Method access structure with the Method such that the Method access structure is contiguous with the code of the Method.

2. (Currently Amended) ~~A~~The method as claimed in Claim 1, wherein the set of frame size determining criteria includes the number of parameter words, the total number of local words, and the number or words of evaluation stack.

3. (Currently Amended) ~~A~~The method as claimed in Claim 1, wherein creating an activation frame for the Method based on the set of frame size determining criteria of the ~~activation stack~~ single frame frame size determining template includes creating a local variable portion, an evaluation stack, and a fixed size frame linkage structure.

4. (Currently Amended) ~~A~~The method as claimed in Claim 1, further comprising associating the Method access structure with a pointer and defining the pointer such that it is an indicator of where code for implementing a Method resides and an indicator for the Method itself.

5. (Currently Amended) ~~A~~The method as claimed in Claim 1, wherein the Method access structure is variably sized.

6. (Currently Amended) A method of increasing the execution speed of invoking a plurality of Methods in an execution device, the plurality of Methods associated with one or more classes, the method comprising:

establishing ~~an activation stack~~ a single frame size determining template ~~with~~ for comparing the single frame size determining template with each of the Methods of the plurality of Methods when they are invoked, the single frame size determining template having a set of frame size determining criteria representative of a predetermined number of words for determining the size of activation frames when creating the activation frames;

for each one of the Methods,

determining whether a word number requirement of the one Method conforms to the frame size determining criteria of the ~~stack~~ single frame size determining template;

conditionally creating a fixed size activation frame regardless of ~~the one Method's~~ an exact stack requirement[s] of the one Method, based on the set of frame size determining criteria of the ~~activation stack~~ single frame size determining template if a word number requirement of the one Method conforms to the set of frame size determining criteria of the ~~activation stack~~ single frame size determining template;

conditionally creating an activation frame to match the one Method's exact stack requirements if the word number requirement of the one Method does not conform to the set of frame size determining criteria of the ~~activation stack~~ single frame size determining template; ~~and~~

spatially optimizing the Method to provide a Method access structure; and

associating [a] the Method access structure with the Method such that the Method access structure is contiguous with the code of the Method;

conditionally creating a Method routing structure external to the Method access structure and pointing to the Method access structure for each class; and

rewriting invocation bytecodes to a form that includes an indication of the Method routing structure.

7. (Currently Amended) ~~A~~The method as claimed in Claim 6, wherein the set of frame size determining criteria includes the number of parameter words, the total number of local words, and the number or words of evaluation stack.

8. (Currently Amended) ~~A~~The method as claimed in Claim 6, wherein creating an activation frame for the Method based on the set of frame size determining criteria of the ~~activation stack~~ single frame size determining template includes creating a local variable portion, an evaluation stack, and a fixed size frame linkage structure.

9. (Currently Amended) ~~A~~The method as claimed in Claim 6, further comprising associating each Method access structure with a pointer and defining the pointer such that it is an indicator of where code for implementing a Method resides and an indicator for the Method itself.

10. (Currently Amended) ~~A~~The method as claimed in Claim 6, further comprising maintaining Method access structures associated with dynamically compiled code in an area of memory separate from Method access structures associated with bytecode.

11. (Currently Amended) ~~A~~The method as claimed in Claim 6, further comprising creating the Method routing structure such that it has one or more misaligned pointers.

12. (Currently Amended) ~~A~~The method as claimed in Claim 11, wherein the

misaligned pointers are used to denote processor executable Method access structures and one or more aligned pointers are used to denote processor non-executable Method access structures.

13. (Currently Amended) ~~A~~The method as claimed in Claim 6, wherein each Method access structure is variably sized.

14. (Currently Amended) An execution system for increasing ~~the~~ an execution speed of invoking Methods of one or more classes, the system comprising:

memory; and

a virtual machine operable to access the memory, to create a representation of at least one of the Methods based on ~~an activation stack~~ a single frame size-determining template ~~with~~ for comparing the single frame size determining template with each of the Methods of the plurality of Methods when they are invoked, the single frame size determining template having a set of frame size determining criteria representative of a predetermined number of words for determining the size of activation frames when creating the activation frames, to conditionally create a representation of at least one of the Methods based on exact stack requirements, to spatially optimize at least one of the Methods to provide a Method access structure and associate a Method access structure contiguous to the representation of each of the Methods, and to create a Method routing structure external to the Method access structure and pointing to the Method access structure for each of the one or more classes in the memory.

15. (Currently Amended) ~~A~~The execution system as claimed in Claim 14, wherein the set of frame size determining criteria includes the number of parameter words, the total number of local words, and the number or words of evaluation stack.

16. (Currently Amended) ~~A~~The execution system as claimed in Claim 14, wherein the ~~activation~~ single frame size determining template includes a local variable portion, an evaluation stack, and a fixed size frame linkage structure.

17. (Currently Amended) ~~A~~The execution system as claimed in Claim 14, wherein the virtual machine is operable to associate a pointer with each Method access structure, the pointer defined such that it is an indicator of where code for implementing a Method resides and an indicator for the Method itself.

18. (Currently Amended) ~~A~~The execution system as claimed in Claim 14, wherein the virtual machine is operable to maintain Method access structures associated with dynamically compiled code in an area of memory separate from Method access structures associated with bytecode.

19. (Currently Amended) ~~A~~The execution system as claimed in Claim 14, wherein the Method routing structure includes one or more misaligned pointers to denote processor executable Method access structures.

20. (Currently Amended) ~~A~~The execution system as claimed in Claim 14, wherein the virtual machine is operable to spatially associate the Method access structure immediately preceding the representation of each of the Methods.